

1. Advanced JavaScript Programming

Adrian Adiaconitei

LINKAcademy

Objective

- ✓ Recapitulare
 - ✓ Obiecte JavaScript
- ✓ Programarea orientată pe obiecte în JavaScript
 - ✓ Clasele și membrii lor

JavaScript

- ✓ ECMAScript 1 (1997)
- ✓ ECMAScript 2 (1998)
- ✓ ECMAScript 3 (1999)
- ✓ **ECMAScript 5 (2009) – compatibila cu 99.9% browsers**
- ✓ ECMAScript 6 (2015)
- ✓ ECMAScript 7 (2016)
- ✓ ECMAScript 8 (2017)
- ✓ ECMAScript 9 (2018)
- ✓ ECMAScript 2019
- ✓ ECMAScript 2020
- ✓ ECMAScript 2021
- ✓ ECMAScript 13 (2022)
- ✓ ECMAScript 14 (March 21, 2023)
- ✓ ECMAScript 15 (March 27, 2024)

<https://tc39.es/ecma262/>

JavaScript

VS code extensii:

JavaScript (ES6) code snippets

Live Server

Prettier - Code formatter

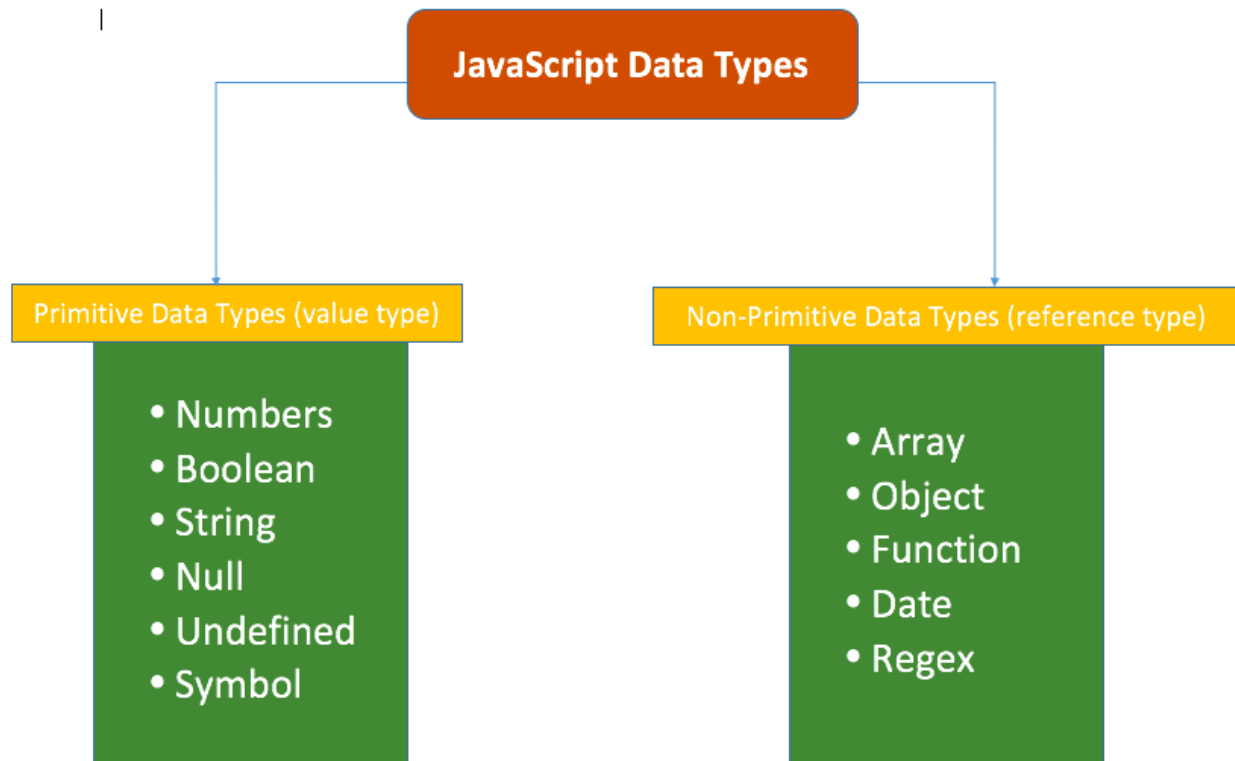
GitHub Copilot

Tabnine AI Autocomplete

Colorize

<https://marketplace.visualstudio.com/vscode>

JavaScript



https://www.w3schools.com/js/js_datatypes.asp

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Symbol

JavaScript - OOP

- ✓ JavaScript este Object Oriented Programming, toate lucrurile din JavaScript sunt obiecte .
- ✓ Un limbaj bazat pe prototip are noțiunea de **obiect prototip**, obiect folosit ca șablon din care să obțină proprietățile inițiale pentru un obiect nou.

Object prototype

- ✓ Fiecare obiect JavaScript are proprietatea **__proto__** în mod implicit, care se referă la **Object prototype**
- ✓ Avem acces la **new Object()**

JavaScript - OOP

this e o variabilă care reprezintă obiectul cu care lucrăm

var x1 = **new** Object(); *// A new Object object*

var x2 = **new** String(); *// A new String object*

var x3 = **new** Number(); *// A new Number object*

var x4 = **new** Boolean(); *// A new Boolean object*

var x5 = **new** Array(); *// A new Array object*

var x6 = **new** RegExp(); *// A new RegExp object*

var x7 = **new** Function(); *// A new Function object*

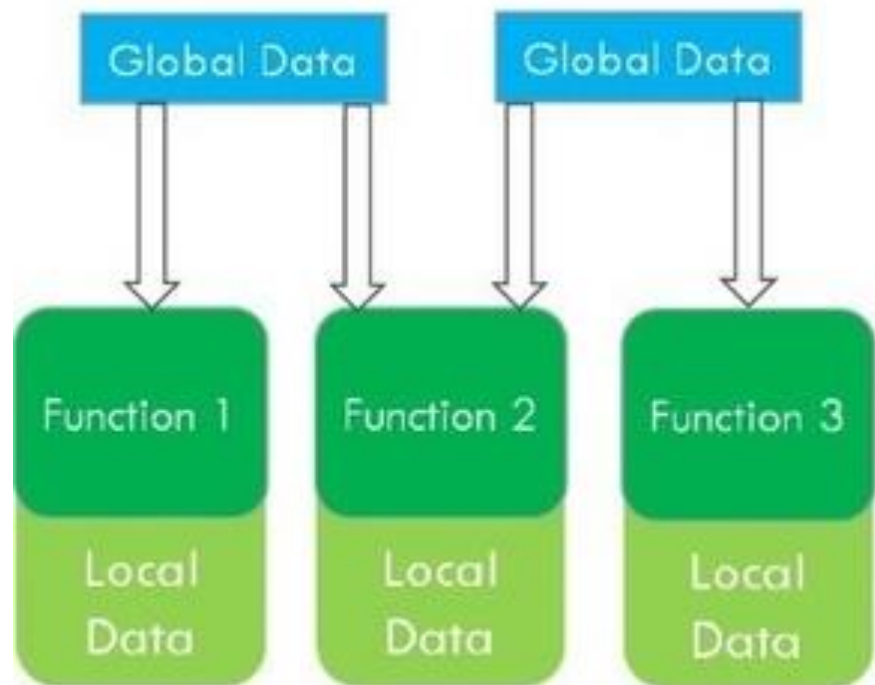
var x8 = **new** Date(); *// A new Date object*

var firstName = **new** String("John");

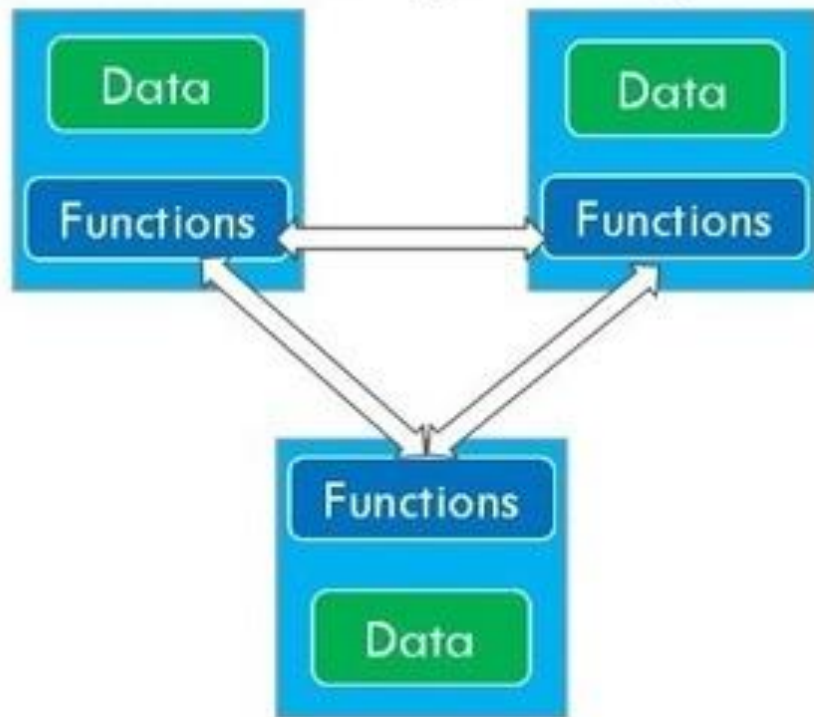
var x = **new** Number(123);

var x = **new** Boolean(false);

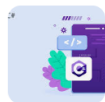
Procedural Oriented Programming



Object Oriented Programming



JavaScript - OOP



C



Ruby



Go



C++



Java



PHP



Swift



C#



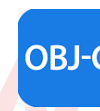
Python



JavaScript



TypeScript



Objective-C

JavaScript - OOP

Avem nevoie de Programare Orientată pe Obiecte?

✓ Scopul programarii orientate pe obiecte este de a încerca să crească flexibilitatea și mentenabilitatea programelor.

-obiectele sunt un tip de date complex ce poate fi definit de programator

-obiectele sunt o colecție de date ce pun la dispoziție și o serie de funcții prin care putem manipula acele date

Cu ajutorul lor putem modela entitățile din lumea reală și să le transpunem în limbajul de programare dorit

JavaScript - OOP

Avantajele OOP

1. Reutilizarea codului
2. Eliminam redundanța datelor —» codul scris într-un singur loc
3. Întreținerea codului
4. Securitate
5. Beneficii de proiectare
6. Productivitate mai bună
7. Depanare/mentenanță ușoară
8. Flexibilitatea polimorfismului

JavaScript - OOP

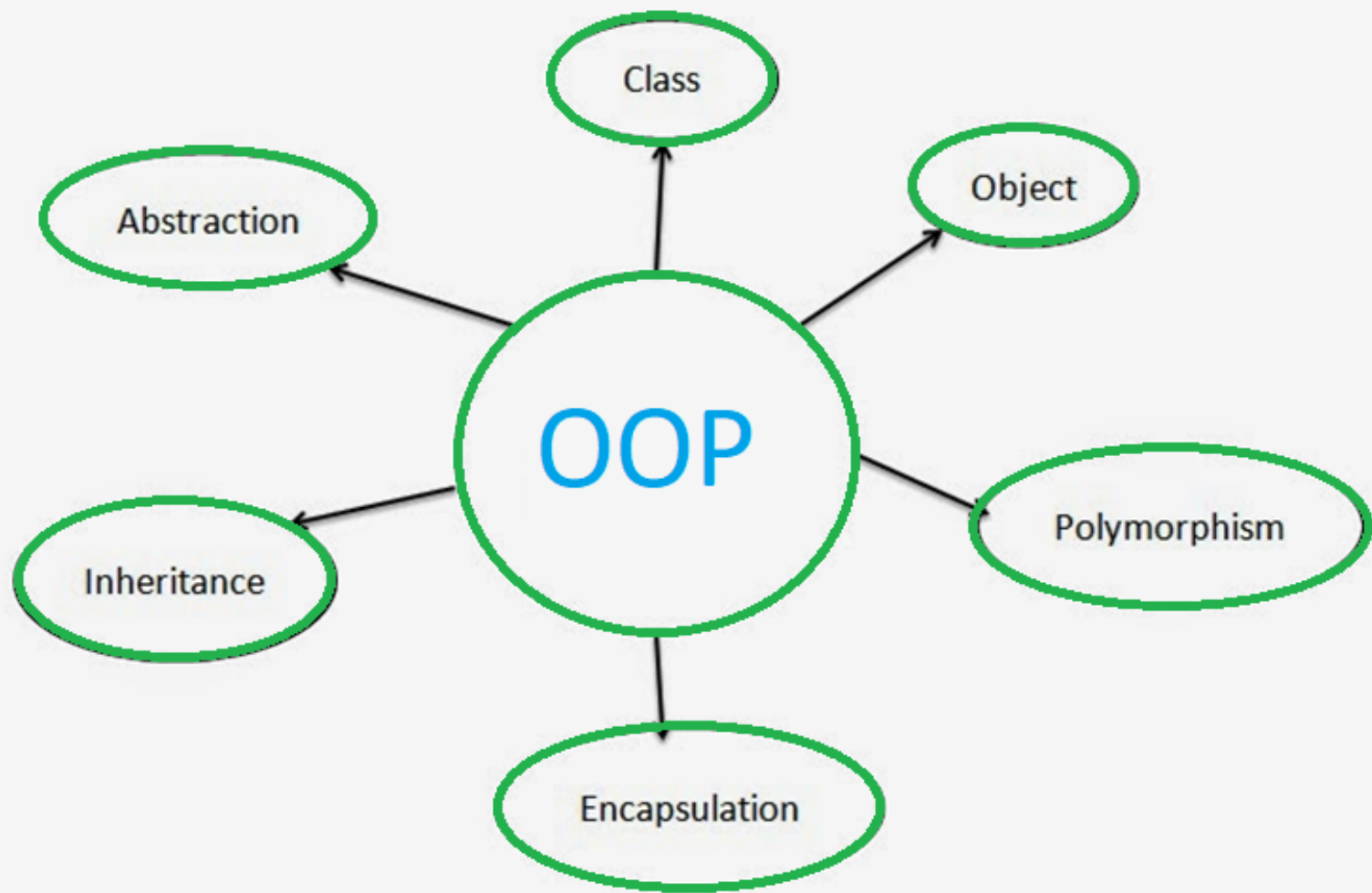
Când să utilizați OOP sau programarea funcțională

✓ **Utilizați OOP atunci când:**

- aveți de-a face cu un sistem complex cu tipuri și relații clar definite.
- Starea aplicației dvs. se schimbă frecvent și trebuie gestionată.
- Preferi o abordare modulară și structurată a organizării codului.

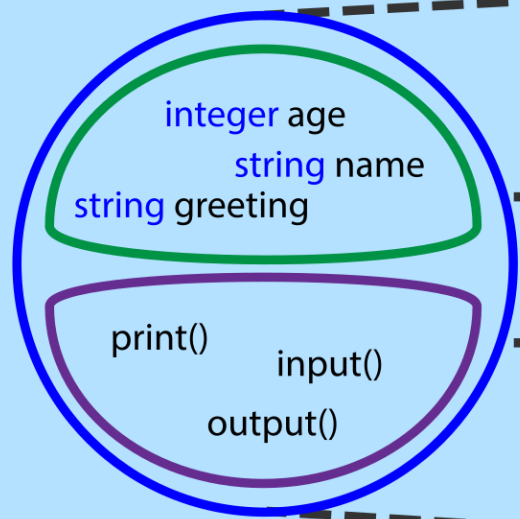
✓ **Utilizați programarea funcțională atunci când:**

- aveți nevoie de un sistem robust, cu operațiuni care nu depind de starea sau nu se modifică foarte des.
- Accentul tău este pe fluxul de date și transformările aplicate acestuia.
- Dorești un cod ușor de testat și de raționat datorită purității sale.

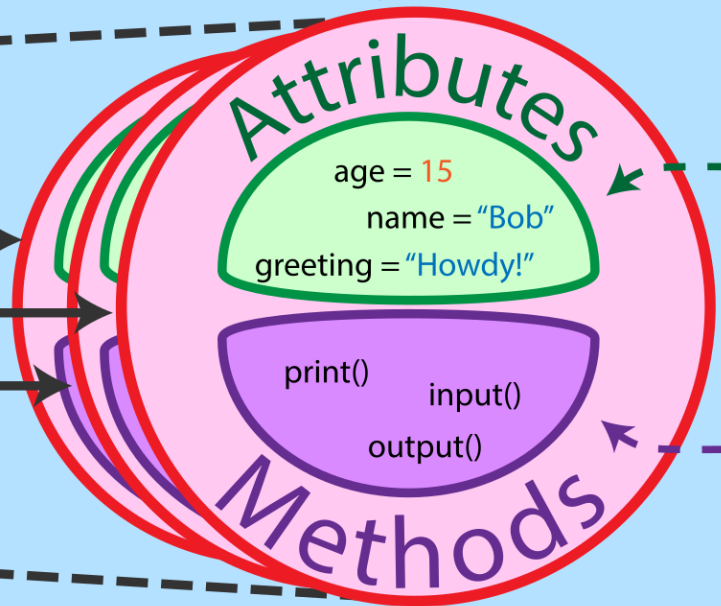


Object Oriented Programming

Class

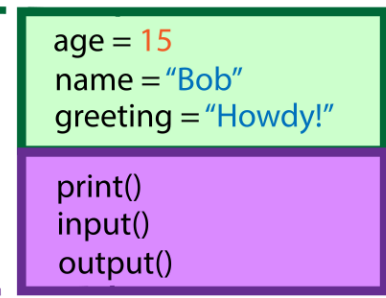


Objects

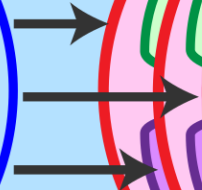


Procedural Programming Equivalent

Variables



Procedures



JavaScript - Crearea obiectelor

1. Definim obiecte folosind metoda clasică: **Object literals**

Ap1.html

```
let masina ={  
    nume:'Dacia';  
    culoare: verde,  
    viteza:180,  
    stop: function (){ this.viteza =0}  
}
```

- ✓ Se folosește doar pentru a crea un singur obiect și pentru a avea acces rapid la proprietățile și metodele obiectului printr-o singură variabilă
- ✓ Această metodă crează obiecte cu structură diferită.
- ✓ Nu am control asupra structurii obiectului, poate fi ușor modificată

JavaScript - Crearea obiectelor

2. definim obiecte folosind metoda **Constructor function cu return** :

Ap2.html

```
function masinaNoua(num, culoare, viteza){  
    let obj={}; // initializam un obiect gol  
    obj.num = num;  
    obj.culoare= culoare;  
    obj.viteza = viteza;  
    obj.stop = function(){ this.viteza =0}  
    return obj;  
}
```

Această metodă crează obiecte cu aceeași structură/prototip.

Dacă vom avea mai multe obiecte de tip masinaNoua, vom avea mai multe metode:stop(), chiar dacă fac același lucru.

JavaScript - Crearea obiectelor

3. definim obiecte folosind metoda **Constructor function**:

Ap3.html

```
function masinaNew(ume, culoare, viteza){ //constructor
    this.ume = ume;
    this.culoare= culoare;
    this.viteza = viteza;
    this.stop = function(){ this.viteza =0}
}
```

Această metodă crează obiecte cu aceeași structură folosind operatorul **new**

JavaScript – Crearea obiectelor

4. definim obiecte folosind metoda ***Object.assign()***

✓ ***Object.assign()*** copiază proprietățile unui obiect într-un alt obiect.

5. definim obiecte folosind metoda ***Object.create()***:

Ap4.html

Ap5.html

Ap6.html

JavaScript - OOP

- ✓ ECMAScript 6 (2015) – a introdus noțiunile de OOP modern: noțiunea **class** (const, let, Arrow function, Object.assign() and Object.is())
- ✓ Caracteristicile limbajelor obiectuale sunt: clase, obiecte, metode, proprietăți

JavaScript - OOP

- ✓ **Clasa** este conceptul de baza in POO ce reunește o colecție de obiecte , fabrică de obiecte
- ✓ **Clasa este un șablon/schiță după care se reproduc obiectele, cunoscute și ca instanțe ale clasei**
- ✓ O clasă va cuprinde definițiile datelor(**atributelor / proprietăților**) și operațiile (**metodele**) ce caracterizează obiectele de o anumită categorie .
- ✓ **Obiectul** este conceptul de bază în programarea orientată obiect(POO) , care asociază datele împreună cu operațiile necesare prelucrării acestora. **Obiectul** sau instanța este **un tip de date**, creat după un anumit model (clasă).
- ✓ **Datele** sunt informații de structură descrise de o mulțime de atribute ale obiectului, iar **operațiile** acționează asupra atributelor obiectului

JavaScript - OOP



objects



Audi

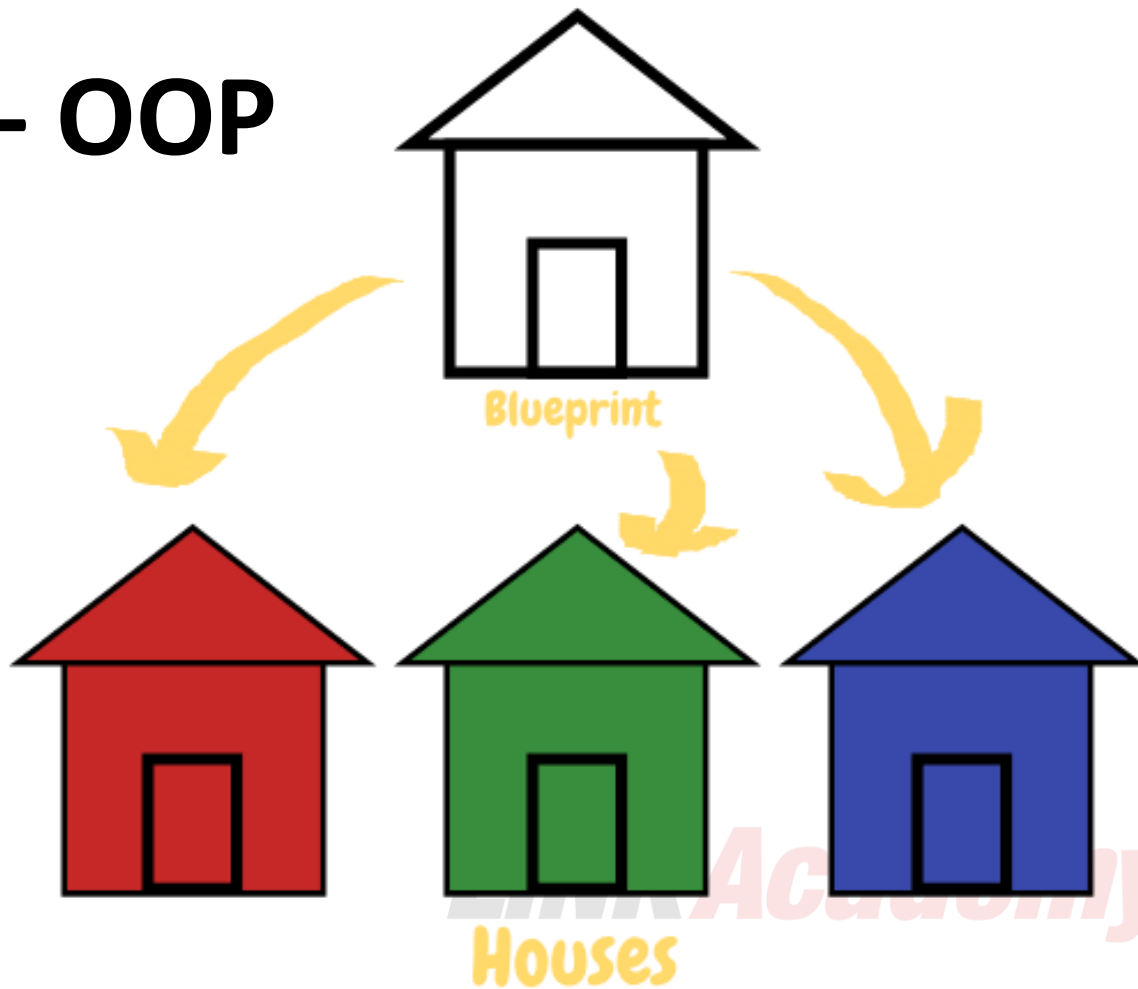


Nissan



Volvo

JavaScript - OOP



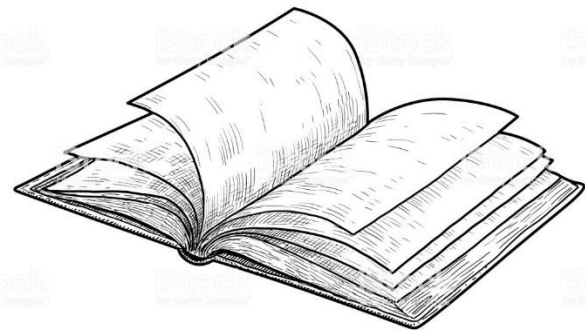
JavaScript - OOP

- ✓ **Datele** definite într-o clasă se mai numesc attribute sau **proprietăți** / variabile, iar operațiile se mai numesc **metode** sau funcții-membru / funcțiile.
- ✓ Proprietățile și metodele formează **membrii** unei clase.
- ✓ ***Definirea unei clase înseamnă crearea unui nou tip de date care apoi poate fi utilizat pentru declararea obiectelor de acest tip.***
- ✓ Fiecare clasă va avea identitate sau nume.
 - ✓ **NumeClasa**
 - ✓ **Proprietăți**
 - ✓ **Metode**

JavaScript - OOP

De exemplu, putem avea **clasa carte**:

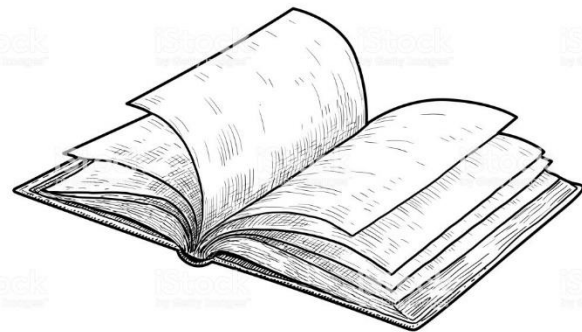
- ✓ **Nume:** `class Carte{}`
- ✓ **Prorpeitati:** titlu, autor, editura, an_apariție, preț, nr_pagini
- ✓ **Metode:** obtine_titlu, obtine_autor, modifica_preț, afiseaza_info, deschide_carte,



JavaScript - OOP

- ✓ Ca să creăm o clasă în JavaScript, creăm un fișier nou (care are același nume ca și clasa pe care vrem să o cream - **Carte.js**) și în el scriem definiția clasei:

```
class Carte{  
}
```



JavaScript - OOP

- ✓ Construirea obiectelor informatice pornind de la clase poartă numele de **instanțiere** sau exemplificare.
- ✓ *Obiectul va fi o instanță a unei clase.*
- ✓ Diferențele dintre obiectele de aceeași clasă se materializează în diferențe între valorile proprietăților.
- ✓ Pentru fiecare obiect este specificat tipul clasei din care provine.
- ✓ Pentru o clasă se pot crea mai multe instanțe ale acesteia.

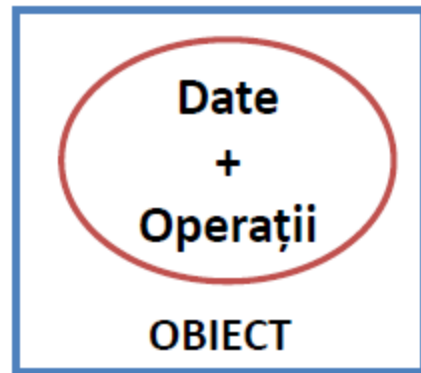
JavaScript - OOP- INSTANȚIEREA

- ✓ După ce facem o clasă, o putem folosi ca să cream **obiecte**

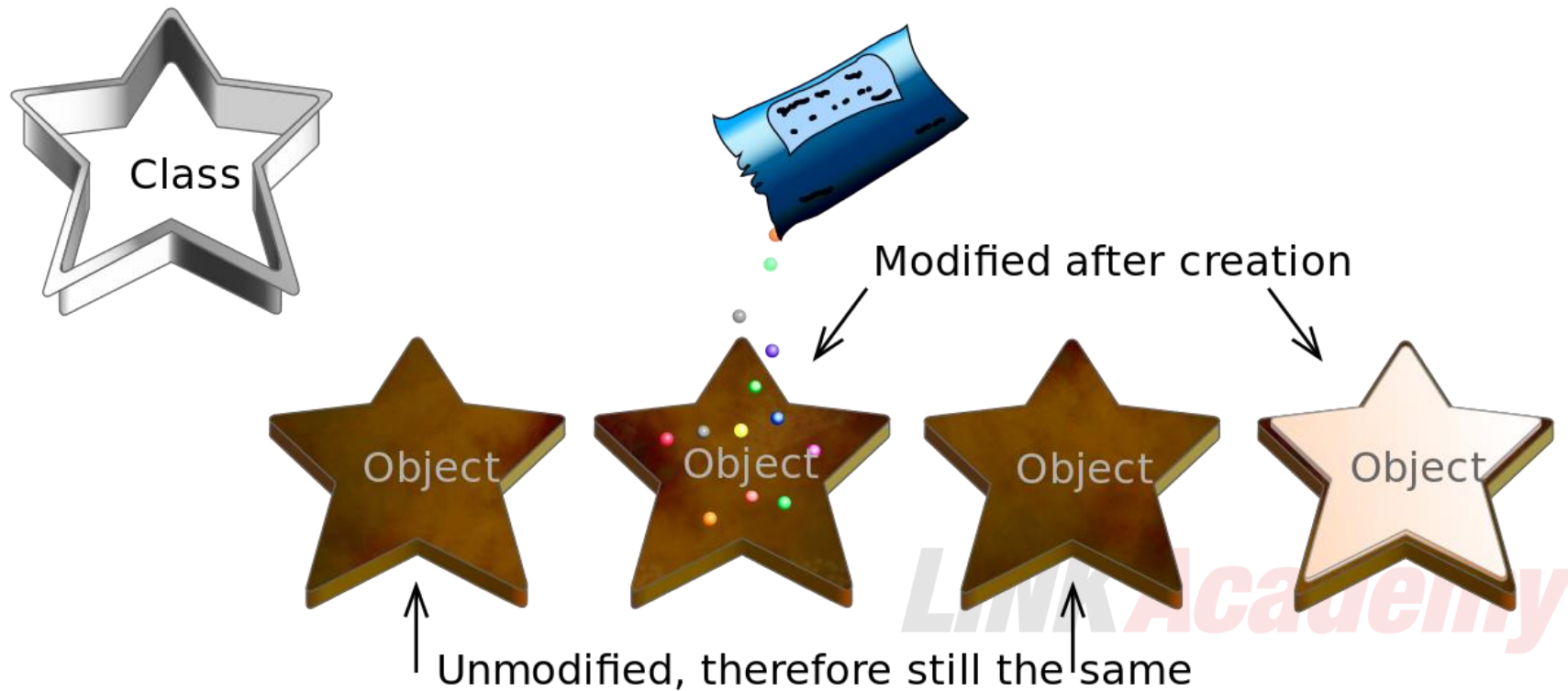
const carte= new Carte(); //creaza un obiect

const carte2= new Carte(); //creaza un alt obiect

LINK



JavaScript - OOP



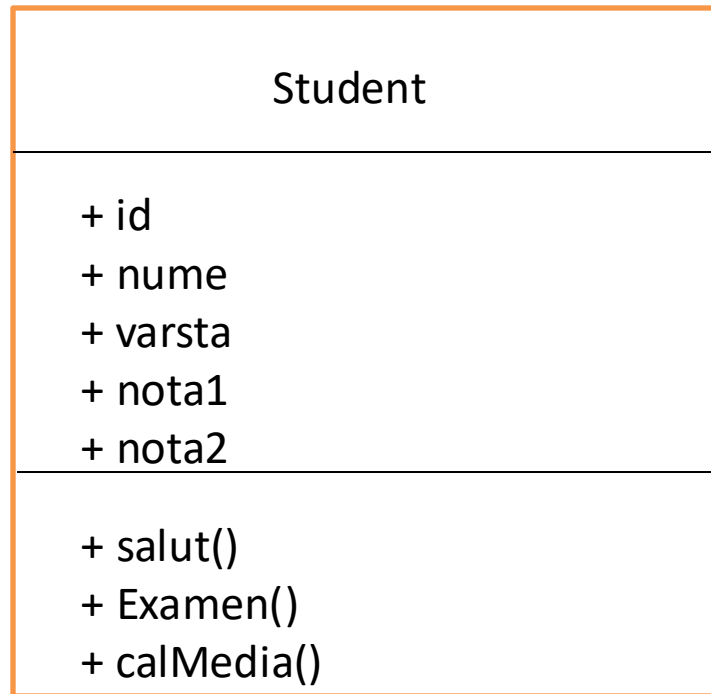
JavaScript - OOP

Aplicația1: Creați 2 obiecte de tip Student, cu structură (membrii obiectului) folosind conceptul de class:

Proprietăți: id, nume, varsta, nota1, nota2, media;

Metode: salut(), examen(), calMedia();

Unified Modeling Language (**UML**)



JavaScript – OOP: Atribute și metode statice

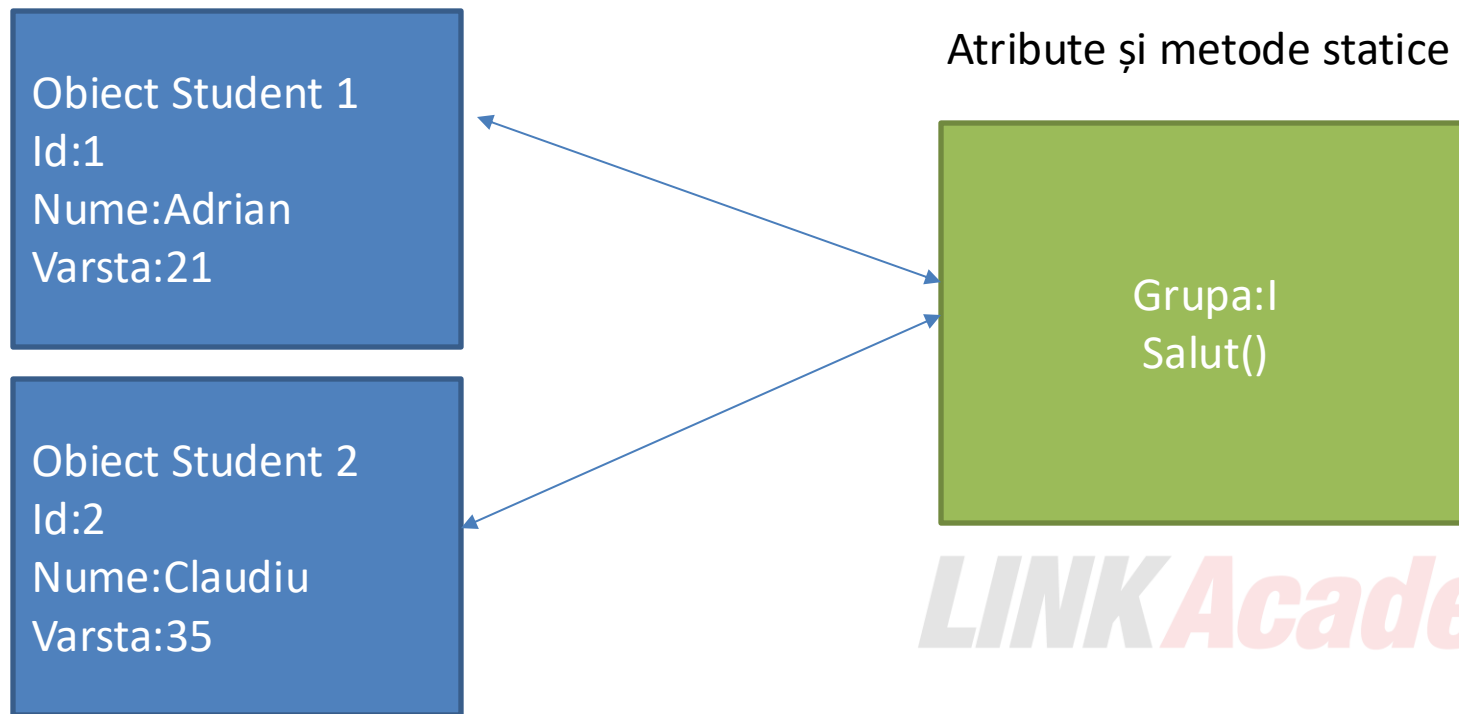
Membrii clasei se împart în: **de instanță** și **statice**.

Atributele și metodele statice, sunt acele care sunt la fel pentru toate obiectele

Pentru a defini o metodă sau proprietate statică folosim numele constructorului în loc de **this**:

```
function Car(model) { //constructor
    this.model = model;
    Car.wheels = 4;
    Car.saySomething = function() {
        alert("Sunt masina");
    }
}
console.log(Car.wheels);
Car.saySomething();
Ap8.html
```

JavaScript – OOP: Atribute și metode statice



JavaScript - OOP- **Attribute / Proprietati**

- ✓ Câmpurile clasei descriu particularități ale clasei. Se mai numesc și **attribute sau proprietăți**. Sunt, de fapt, variabile aflate în cadrul clasei.
- ✓ Attribute și metode statice de la ES6 se definesc folosind : **static**

Ap9.html

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes/static>

JavaScript - OOP- **Attribute / Prorpietati**

De exemplu, attributele unui obiect „carte” pot fi *titlul, autorul, editura, numar pagini, anul aparitiei, pret* atunci **starea unui obiect de tip Carte** ar putea fi :

titlu = Poezii

autor = Mihai Eminescu

editura = Polirom

număr pagini = 125

anul apariției = 2007

preț = 25

JavaScript - OOP- **Attribute / Proprietati**

Aplicația3:

- Să se creze clasa Carte cu attributele specificate mai sus.
- După instanțierea clasei de 3 ori (vom avea 3 obiecte) să se inițializeze attributele cu valorile alese de dumneavoastra.
- Pentru fiecare obiect să se afiseze datele lui.
- Schimbați metodele de instanță în metode **statice**

LINKAcademy

Copiați un obiect într-un alt obiect.

JavaScript - OOP- **Attribute / Proprietati**

Aplicatia3: Generati 3 obiecte de tip carte:

- **titlu** Poezii, Geniu Pustiu, Marile speranțe
- **autor** Mihai Eminescu, Mihai Eminescu, Charles Dickens
- **editura** Polirom, All, Univers
- **an aparitie** 2007, 2005, 2003,
- **pret** 25, 20, 35

JavaScript - OOP - Metode

- ✓ metodele clasei sunt funcții care descriu comportamentul obiectului
- ✓ Se împart în: **intrebari** și **comenzi** (command-query).
- ✓ **comenzi** – spun obiectului să facă ceva
- ✓ **intrebari** – raspund la o întrebare despre obiect(return)
- ✓ pentru a apela o metodă a clasei se folosește: **obj.metoda()**
- ✓ În interiorul metodei referirea la membrii clasei se face cu **this**

```
carte = new Carte();
```

```
carte.setTitlu('Carte de OOP PHP 7');
```

```
carte.setAutor('Autor Anonim');
```

```
carte.getTitlu();
```

```
carte.getAutor();
```

JavaScript – OOP - Metode

- ✓ Un getter este scris ca o metodă fără argument prefixată de cuvântul cheie **get**.
- ✓ Un setter este similar, cu excepția faptului că acceptă un argument (noua valoare fiind atribuită) și este folosit cuvântul cheie **set**.
- ✓

```
class Person {  
    constructor(name) {this._name = name;}  
    get name() {return this._name.toUpperCase();}  
    set name(newName) {  
        this._name = newName;  
    }  
    walk() {console.log(this._name + ' is walking.')}  
}
```

JavaScript – OOP

Aplicația4:

Pentru fiecare atribut al clasei definiți metodele de tip **setter** și de tip **getter**

Definiți ,apoi, și alte metode

Ex. - deschide_cartea(pagina) folosind si referinta catre obiectul curent: this

Definiți pentru toate cărțile o reducere de 20%(folosind **static**)

JavaScript – OOP

Joc 1: Turnurile din Hanoi:

Scopul acestui joc este de a muta piesele din partea stângă pe tija din partea dreaptă. O mutare constă în deplasarea unei piese pe o altă tijă, cu condiția ca în locul destinație ea să fie cea mai mică (ca mărime).



Resurse

- ✓ https://www.w3schools.com/js/js_es6.asp
- ✓ <https://dev.to/bhagatparwinder/classes-not-just-syntactic-sugar-1n0m>
- ✓ <https://coryrylan.com/blog/javascript-es6-class-syntax>
- ✓ <https://www.freecodecamp.org/news/object-oriented-programming-javascript/>
- ✓ <https://exercism.org/tracks/javascript/concepts/classes>